"Express Mail" mailing label number EV 219884547US

Date of Deposit: March 30, 2004

Attorney Docket No.15466US02

SYSTEM AND METHOD FOR PROVIDING DATA STARTING FROM START CODES ALIGNED WITH BYTE BOUNDARIES IN MULTIPLE BYTE WORDS

RELATED APPLICATIONS

[0001] This application claims priority to "SYSTEM AND METHOD FOR PROVIDING DATA STARTING FROM START CODES ALIGNED WITH BYTE BOUNDARIES IN MULTIPLE BYTE WORDS", U.S. Provisional Patent Application, Serial No. 60/541,608, filed February 04, 2004, by Arul Thangaraj,*et al.*, which is incorporated herein by reference for all purposes.

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[0002] [Not Applicable]

[MICROFICHE/COPYRIGHT REFERENCE]

[0003] [Not Applicable]

BACKGROUND OF THE INVENTION

[0004] During the decoding of video data, the video data includes sets of data structures, such as group of pictures, pictures, slices, macroblocks, and blocks. A video decoder decodes the video data, one picture at a time, on a slice by slice basis.

[0005] Start codes indicate the starting points for groups of pictures, pictures, and slices. When a decoding system receives video data for decoding, the decoding system places the video data into a compressed data buffer to await decoding by a video decoder. The compressed data

buffer usually comprises multiple byte data words. For example, the compressed data buffer can comprise 16-byte gigantic words (gwords). The decoder system also creates a start code table the indicates the compressed data buffer data word address for each start code.

[0006] When the video decoder selects a data structure for decoding, the video decoder looks up the start code of the data structure in the start code table to determine the gword address storing the start code. The video decoder then fetches the gwords starting from the address.

[0007] It is noted, however, that the start codes can occur anywhere in the video data and are not necessarily aligned with any particular interval. Accordingly, when the start codes are stored in the compressed data buffer, the start codes do not align with gword boundaries. Therefore, when the video decoder fetches a gword containing a start code, the gword may contain the tail end of another data structure prior to the start code.

[0008] In one scheme, it is possible to assure alignment of start codes with gword boundaries by stuffing zeroes. However, in the case where a data word is 16 bytes, each alignment could add up to 15 bytes, depending on the position of the start code. A transport stream delivers the video data. The transport stream comprises fixed length packets of 188 bytes. As a result, the packets can be fetched with a direct memory access module with 256 byte buffer. However, if a transport packet includes a large number of start codes, the transport packet may require in excess of 256 bytes for storage. If a transport packet requires in excess of 256 bytes for storage, more than one access will be needed to fetch the packet.

[0009] Further limitations and disadvantages of conventional and traditional approaches will become apparent to one of ordinary skill in the art through comparison of such systems with the present invention as set forth in the remainder of the present application with reference to the drawings.

BRIEF SUMMARY OF THE INVENTION

[0010] Described herein is a system and method for system and method for providing data starting from start codes aligned with byte boundaries in multiple byte words.

[0011] In one embodiment, there is described a method for decoding video data. The method comprises writing a start code starting at a byte in a middle portion of a data word in a memory; writing an address associated with the byte in a table; and fetching data from the memory starting from the byte.

[0012] In another embodiment, there is presented a system for decoding video data. The system comprises a memory, a table, and a direct memory access module. The memory comprises a plurality of data words, and stores a start code starting at a byte in a middle portion of a particular one of the data words. The table stores an address associated with the byte. The direct memory access module fetches data from the memory starting from the byte.

[0013] These and other features and advantages of the present invention may be appreciated from a review of the following detailed description of the present invention, along with the accompanying figures in which like reference numerals refer to like parts throughout.

BRIEF DESCRIPTION OF SEVERAL VIEWS OF THE DRAWINGS

[0014] **FIGURE 1** illustrates a block diagram of an exemplary Moving Picture Experts Group (MPEG) encoding process, in accordance with an embodiment of the present invention;

[0015] **FIGURE 2** is a block diagram of an exemplary decoder system in accordance with an embodiment of the present invention;

[0016] **FIGURE 3** is a block diagram describing an exemplary video decoder in accordance with an embodiment of the present invention; and

[0017] **FIGURE 4** is a block diagram of an exemplary direct memory access module in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0018] **FIGURE 1** illustrates a block diagram of an exemplary Moving Picture Experts Group (MPEG) encoding process of video data 101, in accordance with an embodiment of the present invention. The video data 101 comprises a series of frames 103. Each frame 103 comprises two-dimensional grids of luminance Y, 105, chrominance red Cr, 107, and chrominance blue C_b, 109, pixels. The two-dimensional grids are divided into 8x8 blocks, where a group of four blocks or a 16x16 block 113 of luminance pixels Y is associated with a block 115 of chrominance red C_r, and a block 117 of chrominance blue C_b pixels. The block 113 of luminance pixels Y, along with its corresponding block 115 of chrominance red pixels C_r, and block 117 of chrominance blue pixels C_b form a data structure known as a macroblock 111. The macroblock 111 also includes additional parameters, including motion vectors, explained hereinafter. Each macroblock 111 represents image data in a 16x16 block area of the image.

[0019] The data in the macroblocks 111 is compressed in accordance with algorithms that take advantage of temporal and spatial redundancies. For example, in a motion picture, neighboring frames 103 usually have many similarities. Motion causes an increase in the differences between frames, the difference being between corresponding pixels of the frames, which necessitate utilizing large values for the transformation from one frame to another. The differences between the frames may be reduced using motion compensation, such that the transformation from frame to frame is minimized. The idea of motion

compensation is based on the fact that when an object moves across a screen, the object may appear in different positions in different frames, but the object itself does not change substantially in appearance, in the sense that the pixels comprising the object have very close values, if not the same, regardless of their position within the frame. Measuring and recording the motion as a vector can reduce the picture differences. The vector can be used during decoding to shift a macroblock 111 of one frame to the appropriate part of another frame, thus creating movement of the object. Hence, instead of encoding the new value for each pixel, a block of pixels can be grouped, and the motion vector, which determines the position of that block of pixels in another frame, is encoded.

[0020] Accordingly, most of the macroblocks 111 are compared to portions of other frames 103 (reference frames). When an appropriate (most similar, i.e. containing the same object(s)) portion (reference pixels) of a reference frame 103 is found, the difference between the portion of the reference frame 103 and the macroblock 111 are encoded. The difference is known as the prediction error. The location of the reference pixels in the reference frame 103 is recorded as a motion vector. The encoded difference and the motion vector form part of the data structure encoding the macroblock 111. In the MPEG-2 standard, the macroblocks 111 from one frame 103 (a predicted frame) are limited to prediction from portions of no more than two reference frames 103. It is noted that frames 103 used as a reference frame for a predicted frame 103 can be a predicted frame 103 from another reference frame 103.

[0021] The macroblocks 111 representing a frame are grouped into different slice groups 119. The slice group 119 includes the macroblocks 111, as well as additional parameters describing the slice group. Each of the slice groups 119 forming the frame form the data portion of a picture structure 121. The picture 121 includes the slice groups 119 as well as additional parameters that further define the picture 121.

[0022] The pictures are then grouped together as a group of pictures (GOP) 123. The GOP 123 also includes additional parameters further describing the GOP. Groups of pictures 123 are then stored, forming what is known as a video elementary stream (VES) 125. The VES 125 is then packetized to form a packetized elementary sequence. The packetized elementary stream is further packetized into fixed 192-byte (including a 4 byte header, and 188-bytes of data) packets, known as transport packets.

[0023] The transport packets can be multiplexed with other transport packets carrying other content, such as another video elementary stream 125 or an audio elementary stream. The multiplexed transport packets form what is known as a transport stream. The transport stream is transmitted over a communication medium for decoding and displaying.

[0024] Referring now to **FIGURE 2**, there is illustrated a block diagram describing an exemplary decoder system 200 in accordance with an embodiment of the present invention. The decoder system 200 receives a transport stream 205 and stores the transport stream 205 in a transport stream presentation buffer 210. The transport stream presentation

buffer 210 can comprise memory, such as synchronous dynamic random access memory (SD-RAM).

[0025] A transport processor 215 demultiplexes the transport stream 205 into constituent elementary streams. For example the transport stream can comprise any number of video and audio elementary stream constituents. Additionally, the transport processor 215 parses and processes the transport header information from the transport streams stored in the transport stream presentation buffer 210. The constituent audio elementary streams can be provided to an audio decoding section of the decoder system 200.

[0026] A video transport processor 218 writes video elementary stream 125 to a compressed data buffer 220. As noted above, the video elementary stream 125 comprises a hierarchy of various structures, such as GOPs 123, pictures 121, slice groups 119, and macroblocks 111. The starting point of the foregoing is indicated in the video elementary stream 125 by what is known as a start code.

[0027] As the transport processor 218 writes the video elementary stream 125 to the compressed data buffer 220, the transport processor 215 also maintains a start code table 225. The start code table 225 comprises records of start codes and the address in the compressed data buffer 220 storing the start code.

[0028] A video decoder 230 decodes the video elementary stream 125 stored in the compressed data buffer 220. The video decoder 230 decodes the video elementary stream 125 on a picture-by-picture basis, and decodes the pictures on a slice-by-slice basis. The coding of the slice structure

allows for concurrent decoding of multiple slices by hardware in parallel.

[0029] The video decoder 230 can decode a number of slices concurrently. The video decoder 230 begins decoding a slice by fetching the slice from the compressed data buffer 220. To determine the address that the slice group is stored in the compressed data buffer 220, the video decoder 230 looks up the start code for the slice in the start code table 225.

[0030] The compressed data buffer 220 is a memory system comprising multiple byte data words. For example, the compressed data buffer 220 can comprise a DRAM with 16 byte gigantic words (gwords). The start codes align with byte boundaries, but do not necessarily align with gword boundaries. The start code table indicates the byte address in the compressed data buffer 220 where the start code is stored. The video decoder 230 is capable of fetching from the compressed data buffer 220 with byte size granularity.

[0031] Referring now to **FIGURE 3**, there is illustrated a block diagram describing an exemplary video decoder 230 in accordance with an embodiment of the present invention. The video decoder 230 comprises any number of row engines 305. Each row engine 305 can decode a slice concurrently with the other row engines 305.

[0032] The row engines 305 comprise a bitstream extractor 310, a variable length decoder 315, a video direct memory access (DMA) module 320, a master processor 325, and a decompression engine 330. The row engine 305 begins decoding a slice by fetching the slice from the compressed data buffer 220. The master processor 325 looks

up the start code corresponding to the slice in the start code table 225 to find the address of the memory location in the compressed data buffer 220 storing the slice start code. This is the starting address for the slice. The master processor 325 also looks up the next start code in the start code table 225 to find its address in the compressed data buffer 220. The byte preceding this address is the ending address for the slice.

[0033] As noted above, the address is a byte address but does not necessarily correspond to the beginning of a gword. Accordingly, the video DMA 320 is equipped to provide data from the compressed data buffer 220 with byte-size granularity. The master processor 325 commands the DMA 320 to fetch the data in compressed data buffer 220 from the starting address to the ending address.

[0034] Responsive thereto, the video DMA 320 provides the data, and the data is received by the bitstream extractor 310. The bitstream extractor 310 provides the data to the variable length decoder 315. The variable length decoder 315 decodes variable length coded symbols and provides them to the master processor 325. The master processor 325 then provides the video data to the decompression engine 330. The decompression engine 330 decompresses the video data.

[0035] Referring now to **FIGURE 4**, there is illustrated a block diagram of an exemplary direct memory access module 320 in accordance with an embodiment of the present invention. The direct memory access module comprises a state machine 405, a buffer 410, a first mask register 415, a second mask register 420, and an arithmetic logic unit 425.

[0036] As noted above, the start codes fall on byte boundaries but do not necessarily fall on data word boundaries. When the master processor 325 commands the direct memory access module 320 to fetch the data between the address of the start code, and the address preceding the next start code, the direct memory access module 320 retrieves the data word containing the byte where the start code begins, and the data word containing the address preceding the start of the next start code. The foregoing can be achieved by truncating the four least significant bits of the addresses.

[0037] The data words are loaded into the buffer 410. The buffer 410 comprises data words that are equal in length to the data words of the memory. Unless the start code begins on the first byte of a data word, the first data word will contain an ending portion of another data structure prior to the start code. Unless the next start code begins on the first byte of a data word, the last data word will contain a starting portion of another data structure, starting from the next start code.

[0038] The direct memory access module 320 discards the starting and ending portions of other data structures with the first mask register 415 and the second mask register 420. The first mask register 415 is equal in length to a data word in the memory. The state machine 405 writes a pattern to the first mask register 415, wherein each byte in the first mask register 415 corresponding to a byte position that is less than the four least significant bits of the starting address are loaded with a $0, and wherein each byte in the first mask register 415 corresponding to a byte position that is equal or greater than the four least

significant bits of the starting address are loaded with a $F.

[0039] The second mask register 420 is also equal in length to a data word in the memory. The state machine 405 writes a pattern to the second mask register 420, wherein each byte in the second mask register 420 that corresponds to a byte position that is less than or equal to the four least significant bits of the ending address is loaded with a $F, and wherein each byte in the second mask register 420 corresponding to a byte position that is greater than the four least significant bits of the ending address is loaded with a $0.

[0040] The state machine 405 then causes the arithmetic logic unit 425 to perform a bit by bit logical AND operation between the first data word in the buffer 410 and the first mask register 415. The foregoing results in zeroing the ending portion of the previous data structure that precedes the start code. Similarly, the state machine 405 then causes the arithmetic logic unit 425 to perform a bit by bit logical AND operation between the last data word in the buffer and the second mask register 420. The foregoing results in zeroing the starting portion of the next data structure. The direct memory access module 320 then provides the contents of the buffer 410 to the bit stream extractor 310.

[0041] One embodiment of the present invention may be implemented as a board level product, as a single chip, application specific integrated circuit (ASIC), or with varying levels integrated on a single chip with other portions of the system as separate components. The degree of integration of the system will primarily be determined

by speed and cost considerations. Because of the sophisticated nature of modern processors, it is possible to utilize a commercially available processor, which may be implemented external to an ASIC implementation of the present system. Alternatively, if the processor is available as an ASIC core or logic block, then the commercially available processor can be implemented as part of an ASIC device with various functions implemented as firmware.

[0042] While the present invention has been described with reference to certain embodiments, it will be understood by those skilled in the art that various changes may be made and equivalents may be substituted without departing from the scope of the present invention. In addition, many modifications may be made to adapt a particular situation or material to the teachings of the present invention without departing from its scope. Therefore, it is intended that the present invention not be limited to the particular embodiment disclosed, but that the present invention will include all embodiments falling within the scope of the appended claims.